Plotting and Visualization

CS 3753 Data Science

Prof. Weining Zhang

Topics

- Figure and subplots
- Color, marks, line style
- Ticks, labels, and legends
- Plot functions
- Visualize Data

Overview Plotting and Visualization

- One important task in Explorative Data Analysis (EDA) is to plot and visualize data.
- Python provides several packages to plot data.
- The matplotlib package is one of those packages. It is very similar to the plotting functions provided by MatLab and R.
- The pandas package also come with plotting functions that are based on matplotlib
- Another package is seaborn that plot statistical features of the data

We introduce the basic plotting features here.

In [1]:
```
%matplotlib inline
from __future__ import division
from numpy.random import randn
import numpy as np
np.random.seed(12345)
np.set_printoptions(precision=4)
import pandas as pd
from pandas import Series, DataFrame
import os
```

In [ ]:
```
%pwd
```

The Pyplot API

The matplotlib.pyplot (https://matplotlib.org/devdocs/api/pyplot_summary.html) module contains functions to generate many kinds of plots quickly. For examples that showcase the use of the matplotlib.pyplot module, see the Pyplot tutorial (../tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py) or the Pyplot Examples (../gallery/index.html#pyplots-examples).

In [2]:
```
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
```

In [ ]:
```
# Plot 50 random numbers, x-axis is a range, y-axis is the value, da
shline, black
plt.plot(randn(50).cumsum(), 'k--')
```

See matplotlib.pyplot.plot (https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot) for details

Figures and Subplots

- Figure Class (https://matplotlib.org/devdocs/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure)
- Axes Class (https://matplotlib.org/api/axes_api.html)

- A figure is an object that can contain one or more subplots.
- The subplots are organized into grid.
- Each subplot is represented by an Axes object, which provides functions to draw the subplot

Create Figure and Subplots

```
plt.figure(): Create a figure
plt.subplots(nrows, mcols, ...): Create a fig and a list of Axes
                                 representing n x m subplots
fig.add_subplot(nrows, ncols, index, ...)
   : Returns an Axes at index position in grid as a subplot.
     For 2 x 3 grid, the index positions are
         subplot1 subplot2 subplot3
         subplot4 subplot5 subplot6
```

In [ ]:
```
# Create a fig
fig = plt.figure()
# Add 1st subplot on a grid with 2x2 cell per plot
# Create an Axes object
ax1 = fig.add_subplot(2, 2, 1)
fig
```

In [ ]:
```
# Add two more subplots
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 4)
fig
```

Axes Objects

Axes object has functions to produce many different type of plots:

- Basic
- Spans
- Spectral
- Statistics
- Binned
- Contours
- Array
- Unstructured Triangles
- Text and Annotations
- Fields

Example Plot Functions

```
ax.plot(x, y, ...)   : Plot y against x as line
ax.bar(x, height,
          width=...) : Plot a bar chart
ax.hist(x, bins=...) : Plot a histogram of an arry of values
ax.scatter(x, y, ...): Plot scatter plot of y vs x
                       with varying marker size and/or color
```

- Also

```
plt.plot(x, y,...), plt.hist(x,...), plt.bar(x,...), ...
```

In [ ]:

```
# plot in sub-figure 1 the histogram of 100 random numbers in 20 bin
s,
# semi-transparent
_=ax1.hist(randn(100), bins=20, color='k', alpha=0.3)
# plot a scatter plot of 30 randomly generated 2-d points
x = np.arange(30)
y = np.arange(30) + 3 * randn(30)
ax2.scatter(x, y)
# plot a dashed line of 50 random numbers
ax3.plot(randn(50).cumsum(), 'k--')
fig
```

In [ ]:
```
plt.close('all')
```

In [ ]:
```
# Use subplots() to create a fig with 6 sub-figures
fig, axes = plt.subplots(2, 3)
axes
```

Adjusting the Spacing Around Subplots

```
plt.subplots_adjust(left=..., bottom=...,
                    right=..., top=...,
                    wspace=..., hspace=...)

- Default: left = 0.125, right = 0.9,
           bottom = 0.1, top = 0.9
           wspace = 0.2   # width of white space between subplots
           hspace = 0.2   # height of white space between subplots
```

In [ ]:
```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
# plat histograms in each axis
for i in range(2):
    for j in range(2):
        axes[i, j].hist(randn(500), bins=50, color='k', alpha=0.5)
plt.subplots_adjust(wspace=0, hspace=0)
```

Colors, Markers, and Line Styles

The plot function can take parameters that describe the colors, markers, and styles of various components of a plot. The descriptions is given as predefined strings.

character description

```
'-'    solid line style
'--'   dashed line style
'-.'   dash-dot line style
':'    dotted line style
'.'    point marker
','    pixel marker
'o'    circle marker
'v'    triangle_down marker
'^'    triangle_up marker
'<'    triangle_left marker
'>'    triangle_right marker
'1'    tri_down marker
'2'    tri_up marker
'3'    tri_left marker
'4'    tri_right marker
's'    square marker
'p'    pentagon marker
'*'    star marker
'h'    hexagon1 marker
'H'    hexagon2 marker
'+'    plus marker
'x'    x marker
'D'    diamond marker
'd'    thin_diamond marker
'|'    vline marker
'_'    hline marker
```

The following color abbreviations are supported:

```
‘b’ blue
‘g’ green
‘r’ red
‘c’ cyan
‘m’ magenta
‘y’ yellow
‘k’ black
‘w’ white
```

In [ ]:
```
# Create a figure
plt.figure()
```

In [ ]:
```
plt.plot(randn(30).cumsum(), 'r^-.')
```

In [ ]:
```
plt.close('all')
```

In [ ]:
```
# Plot multiple lines in one figure
data = randn(30).cumsum()
plt.plot(data, 'g--', label='Default')

# drawstyle= [‘default’ | ‘steps’ | ‘steps-pre’ | ‘steps-mid’ | ‘ste
ps-post’]
plt.plot(data, 'y-.', drawstyle='steps-pre', label='steps-post')
plt.legend(loc='best')
```

See matplotlib.pyplot.plot (https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot) for details

Ticks, Labels, and Legends

Setting Title, Axis Labels, Ticks, and Tick Labels

Each Axes object can define ticks for x and y axises. The ticks can be set on specific positions, with its own label. The label font size and direction can also be set.

```
ax.set_title(title ...)
ax.set_xlabel(label ..)
ax.set_xticks(tickes, ...)
ax.set_xticklabels(labels, ...)
```

In [ ]:
```
fig = plt.figure()
# use an Axes object to contol the ploting features
ax = fig.add_subplot(1, 1, 1)

ticks = ax.set_xticks([0, 250, 500, 750, 1000])
labels = ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'],
                            rotation=60, fontsize='small')
ax.set_title('A plot of cumsum for 1000 random numbers')
ax.set_xlabel('Stages')

ax.plot(randn(1000).cumsum())
```

Adding Legends

```
ax.plot(array, ..., label=...)
ax.legend(loc=...)
```

In [ ]:
```
fig = plt.figure(); ax = fig.add_subplot(1, 1, 1)
# plot three lines from different sets of data
ax.plot(randn(1000).cumsum(), 'k', label='one')
ax.plot(randn(1000).cumsum(), 'g--', label='two')
ax.plot(randn(1000).cumsum(), 'r.', label='three')

ax.legend(loc='best')
```

Annotations and Drawing on a Subplot

When a curve is plotted, the user may add notes and markers to the figure to indicate interesting points on the curve

```
ax.annotate(...) : specify label and label placement
```

In following example, we add downward arrows and markers to a stock index plot

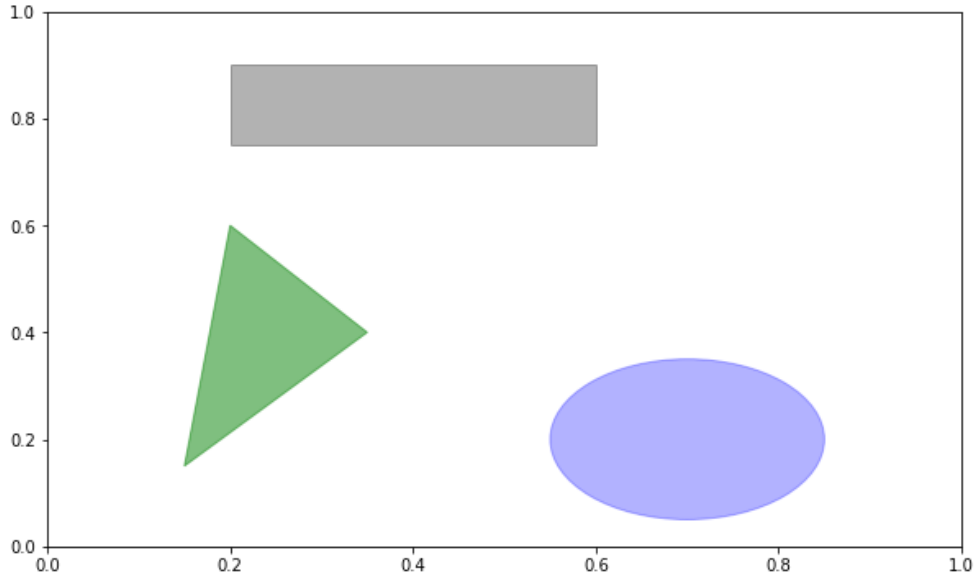
In [ ]:
```
from datetime import datetime

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

# read SP500 index data into a DataFrame
data = pd.read_csv('ch08/spx.csv', index_col=0, parse_dates=True)
spx = data['SPX']
print("data =\n", data[:5])
print("spx =\n", spx[:5])

# Plot SP500 index data
spx.plot(ax=ax, style='k-')

crisis_data = [
    (datetime(2007, 10, 11), 'Peak of bull market'),
    (datetime(2008, 3, 12), 'Bear Stearns Fails'),
    (datetime(2008, 9, 15), 'Lehman Bankruptcy')
]

for date, label in crisis_data:
    # add annotation in the plot, set coordinates for text and arrow
    ax.annotate(label, xy=(date, spx.asof(date) + 100),
                xytext=(date, spx.asof(date) + 230),
                arrowprops=dict(facecolor='black'),
                horizontalalignment='left', verticalalignment='top')

# Zoom in on 2007-2010
ax.set_xlim(['1/1/2007', '1/1/2011'])
ax.set_ylim([600, 1800])

ax.set_title('Important dates in 2008-2009 financial crisis')
```

Drawing Graphics

```
plt.Rectangle(),
plt.Circle(),
plt.Polygon() : Create graphical objects
ax.add_patch(): draw graphical objects
```

In [3]:
```
# Draw shapes in a plot
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

rect = plt.Rectangle((0.2, 0.75), 0.4, 0.15, color='k', alpha=0.3)
circ = plt.Circle((0.7, 0.2), 0.15, color='b', alpha=0.3)
pgon = plt.Polygon([[0.15, 0.15], [0.35, 0.4], [0.2, 0.6]],
                   color='g', alpha=0.5)

ax.add_patch(rect)
ax.add_patch(circ)
ax.add_patch(pgon)
```

Out[3]:
```
<matplotlib.patches.Polygon at 0x10fe9b208>
```

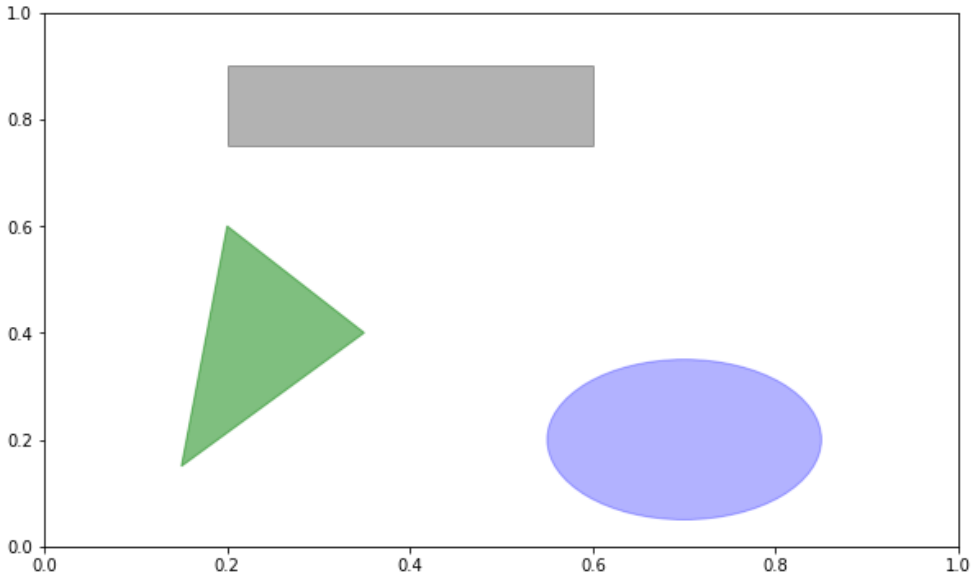


Saving Plots to Files

- Plotted figures can be saved into an image file and read back in later
- Need specialized packages

In [4]:
```
fig
```

Out[4]:



In [5]:
```
fig.savefig('figpath.svg')
```

In [6]:
```
fig.savefig('figpath.png', dpi=100, bbox_inches='tight')
```

In [7]:
```
# Save plot into an in-memory file
from io import BytesIO
buffer = BytesIO()
plt.savefig(buffer, format='png')
plot_data = buffer.getvalue()
```

```
<Figure size 720x432 with 0 Axes>
```

In [9]:
```
plot_data
```

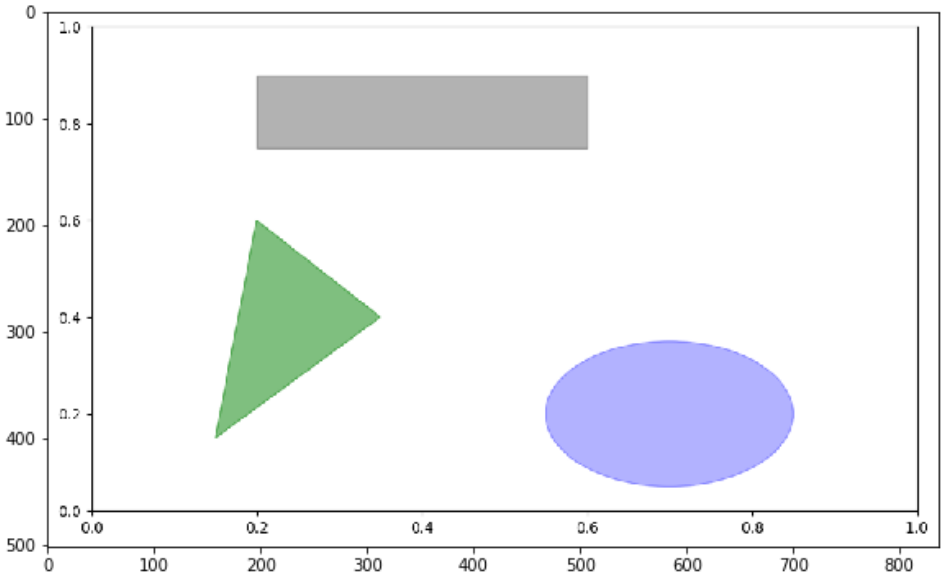
Out[9]:
```
b"\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x02\xd0\x00\x00\x0
1\xb0\x08\x06\x00\x00\x00\x80KR[\x00\x00\x00\x04sBIT\x08\x08\x08
\x08|\x08d\x88\x00\x00\x00\tpHYs\x00\x00\x0b\x12\x00\x00\x0b\x12
\x01\xd2\xdd~\xfc\x00\x00\x009tEXtSoftware\x00matplotlib version
2.2.2, http://matplotlib.org/\x86\x9f\xd4\x0b\x00\x00\x08ZIDATx\
x9c\xed\xd6\xc1\r\xc0 \x10\xc0\xb0\xd2\xfdw>V /\x84dO\x90g\xd6\x
cc\xcc\x07\x00\x00\x1c\xf9o\x07\x00\x00\xc0K\x0c4\x00\x00\x04\x0
6\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\x00\x80\xc0@\x03\x00
@`\xa0\x01\x00 0\xd0\x00\x00\x10\x18h\x00\x00\x08\x0c4\x00\x00\x
04\x06\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\x00\x80\xc0@\x0
3\x00@`\xa0\x01\x00 0\xd0\x00\x00\x10\x18h\x00\x00\x08\x0c4\x00\
x00\x04\x06\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\x00\x80\xc
0@\x03\x00@`\xa0\x01\x00 0\xd0\x00\x00\x10\x18h\x00\x00\x08\x0c4
\x00\x00\x04\x06\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\x00\x
80\xc0@\x03\x00@`\xa0\x01\x00 0\xd0\x00\x00\x10\x18h\x00\x00\x08
\x0c4\x00\x00\x04\x06\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\
x00\x80\xc0@\x03\x00@`\xa0\x01\x00 0\xd0\x00\x00\x10\x18h\x00\x0
0\x08\x0c4\x00\x00\x04\x06\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81
\x06\x00\x80\xc0@\x03\x00@`\xa0\x01\x00 0\xd0\x00\x00\x10\x18h\x
00\x00\x08\x0c4\x00\x00\x04\x06\x1a\x00\x00\x02\x03\r\x00\x00\x8
1\x81\x06\x00\x80\xc0@\x03\x00@`\xa0\x01\x00 0\xd0\x00\x00\x10\x
18h\x00\x00\x08\x0c4\x00\x00\x04\x06\x1a\x00\x00\x02\x03\r\x00\x
00\x81\x81\x06\x00\x80\xc0@\x03\x00@`\xa0\x01\x00 0\xd0\x00\x00\
x10\x18h\x00\x00\x08\x0c4\x00\x00\x04\x06\x1a\x00\x00\x02\x03\r\
x00\x00\x81\x81\x06\x00\x80\xc0@\x03\x00@`\xa0\x01\x00 0\xd0\x00
\x00\x10\x18h\x00\x00\x08\x0c4\x00\x00\x04\x06\x1a\x00\x00\x02\x
03\r\x00\x00\x81\x81\x06\x00\x80\xc0@\x03\x00@`\xa0\x01\x00 0\xd
0\x00\x00\x10\x18h\x00\x00\x08\x0c4\x00\x00\x04\x06\x1a\x00\x00\
x02\x03\r\x00\x00\x81\x81\x06\x00\x80\xc0@\x03\x00@`\xa0\x01\x00
0\xd0\x00\x00\x10\x18h\x00\x00\x08\x0c4\x00\x00\x04\x06\x1a\x00\
x00\x02\x03\r\x00\x00\x81\x81\x06\x00\x80\xc0@\x03\x00@`\xa0\x01
\x00 0\xd0\x00\x00\x10\x18h\x00\x00\x08\x0c4\x00\x00\x04\x06\x1a
\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\x00\x80\xc0@\x03\x00@`\xa
0\x01\x00 0\xd0\x00\x00\x10\x18h\x00\x00\x08\x0c4\x00\x00\x04\x0
6\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\x00\x80\xc0@\x03\x00
@`\xa0\x01\x00 0\xd0\x00\x00\x10\x18h\x00\x00\x08\x0c4\x00\x00\x
04\x06\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\x00\x80\xc0@\x0
3\x00@`\xa0\x01\x00 0\xd0\x00\x00\x10\x18h\x00\x00\x08\x0c4\x00\
x00\x04\x06\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\x00\x80\xc
0@\x03\x00@`\xa0\x01\x00 0\xd0\x00\x00\x10\x18h\x00\x00\x08\x0c4
\x00\x00\x04\x06\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\x00\x
80\xc0@\x03\x00@`\xa0\x01\x00 0\xd0\x00\x00\x10\x18h\x00\x00\x08
\x0c4\x00\x00\x04\x06\x1a\x00\x00\x02\x03\r\x00\x00\x81\x81\x06\
```

Read and Display Images from Image Files

Need to use packages for image read and display

In [10]:
```
import matplotlib.image as mpimg
# read image into a NUmPy narray
img = mpimg.imread('figpath.png')
plt.imshow(img)
```

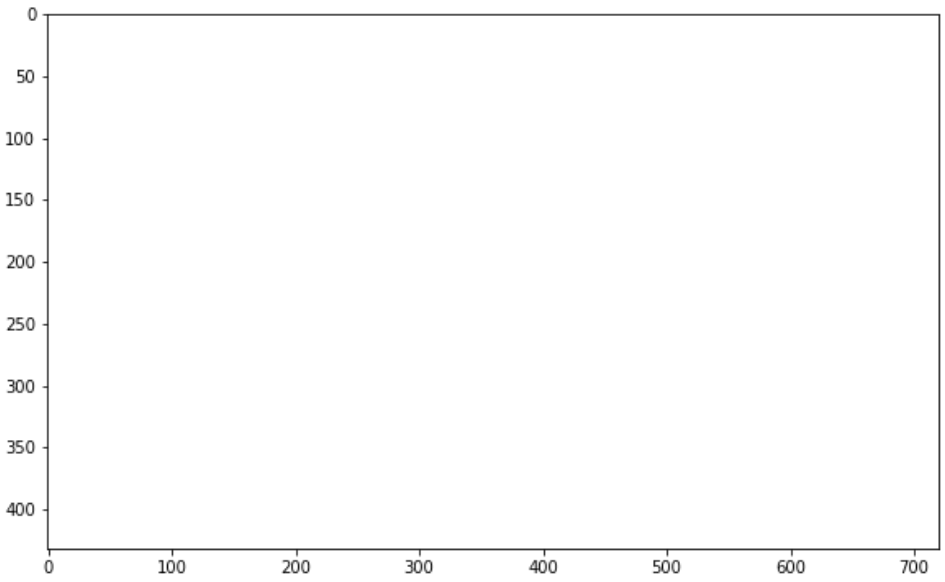
Out[10]: <matplotlib.image.AxesImage at 0x1104317f0>



In [11]:
```
from PIL import Image
buffer.seek(0)
im = Image.open(buffer)
print(im.format, im.size, im.mode)
# Need to install XV in order to show the image correctly
# im.show()
plt.imshow(im)
```

```
PNG (720, 432) RGBA
```

Out[11]: <matplotlib.image.AxesImage at 0x116f573c8>



Matplotlib Configuration

The configuration of current figure can be modified with the plt.rc() function

In [ ]:
```
plt.rc('figure', figsize=(10, 10))
plt.plot(randn(50).cumsum(), 'k--')
```

In [ ]:
```
plt.rc('figure', figsize=(2, 2))
plt.plot(randn(50).cumsum(), 'k--')
```

In [ ]:
```
plt.rc('figure', figsize=(6, 6))
```

In [ ]:
```
plt.close('all')
```

Plotting Functions in Pandas

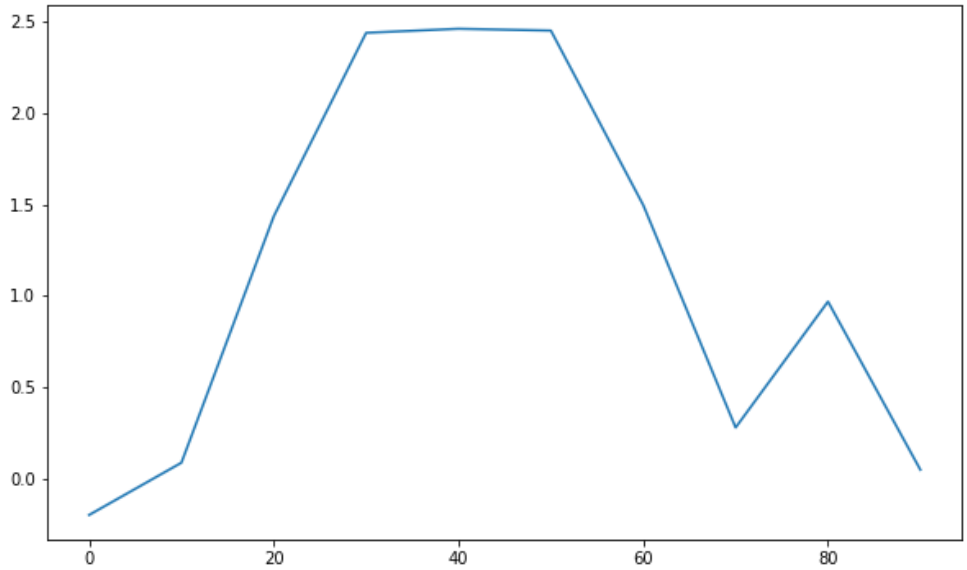
```
df.plot(kind=...)
kind :
 'bar' : vertical bar plot
 'barh' : horizontal bar plot
 'hist' : histogram
 'box' : boxplot
 'kde' : Kernel Density Estimation plot
 'pie' : pie plot
 'scatter' : scatter plot
```

Line Plots

```
kind='line' : default
```

In [13]:
```
df = DataFrame(np.random.randn(10, 4).cumsum(0),
               columns=['A', 'B', 'C', 'D'],
               index=np.arange(0, 100, 10))
df.plot()
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x117366b70>

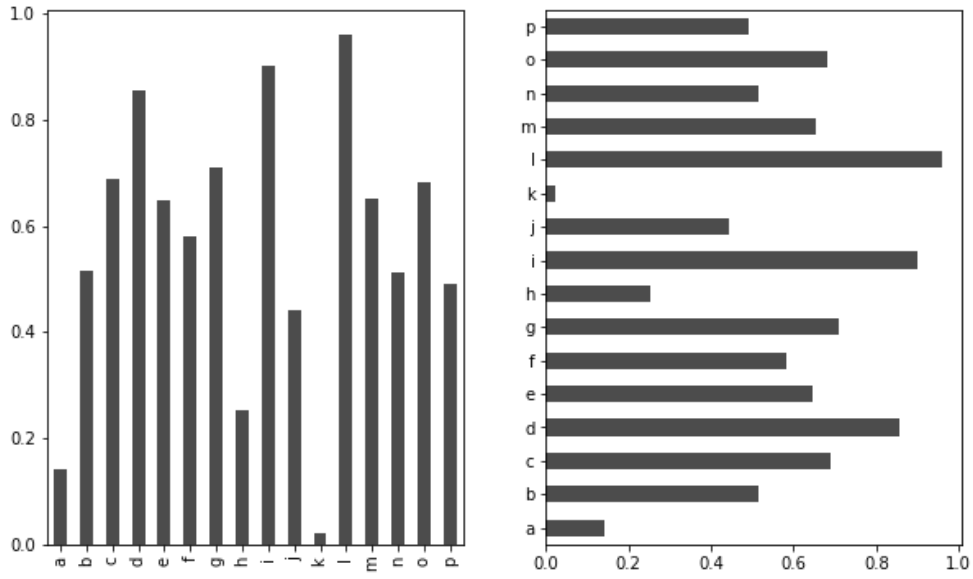


Bar Plots

```
kind='bar'  : vertical bars
    ='barh' : horizontal bars
```

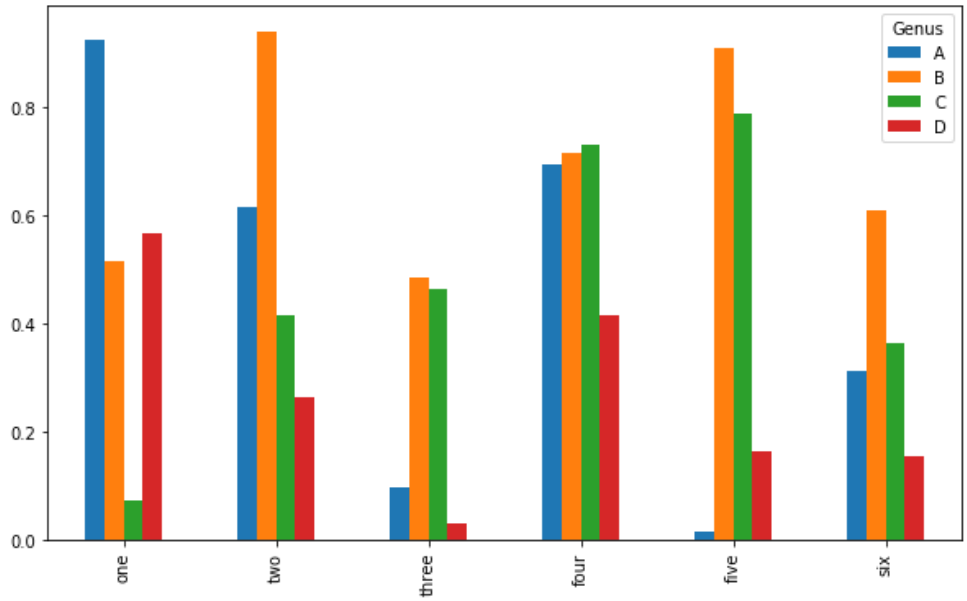
In [14]:
```
fig, axes = plt.subplots(1, 2)
data = Series(np.random.rand(16), index=list('abcdefghijklmnop'))
data.plot(kind='bar', ax=axes[0], color='k', alpha=0.7)
data.plot(kind='barh', ax=axes[1], color='k', alpha=0.7)
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x11759a2b0>



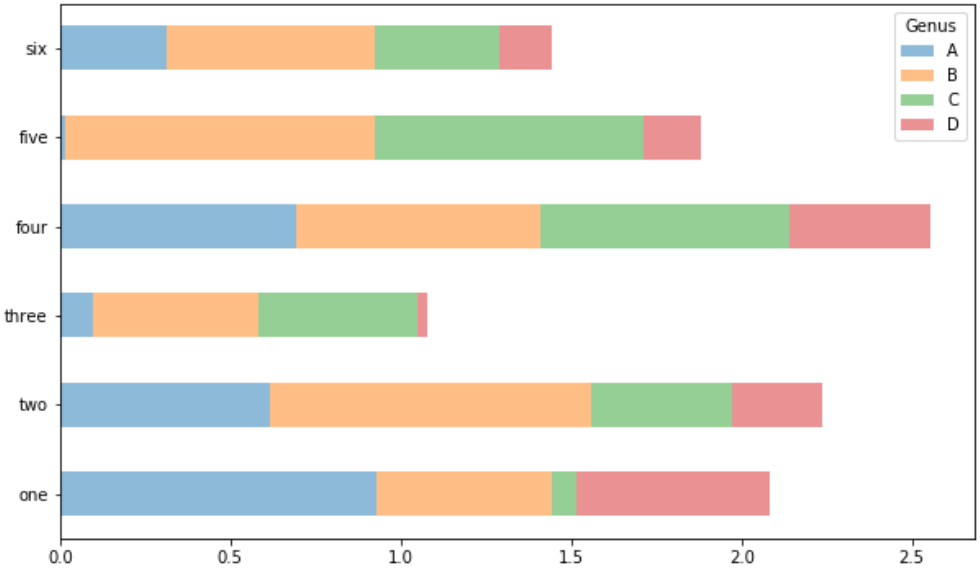
In [15]:
```
df = DataFrame(np.random.rand(6, 4),
               index=['one', 'two', 'three', 'four', 'five', 'six'],
               columns=pd.Index(['A', 'B', 'C', 'D'], name='Genus'))
df.plot(kind='bar')
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x11762bc18>



In [16]:
```
df.plot(kind='barh', stacked=True, alpha=0.5)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1174875c0>



Visualize tips Dataset

- Read in data set, create a crossover table showing counts of tables by day and party size
- Normalize data by day total number of tables
- Plot bar chart comparing days and party sizes

In [17]:
```
tips = pd.read_csv('ch08/tips.csv')
print(tips)
#party_counts = pd.crosstab(tips.day, tips.size)
party_counts = pd.crosstab(tips['day'], tips['size'])
party_counts
```

```
     total_bill   tip     sex smoker   day    time  size
0          16.99  1.01  Female     No   Sun  Dinner     2
1          10.34  1.66    Male     No   Sun  Dinner     3
2          21.01  3.50    Male     No   Sun  Dinner     3
3          23.68  3.31    Male     No   Sun  Dinner     2
4          24.59  3.61  Female     No   Sun  Dinner     4
5          25.29  4.71    Male     No   Sun  Dinner     4
6           8.77  2.00    Male     No   Sun  Dinner     2
7          26.88  3.12    Male     No   Sun  Dinner     4
8          15.04  1.96    Male     No   Sun  Dinner     2
9          14.78  3.23    Male     No   Sun  Dinner     2
10         10.27  1.71    Male     No   Sun  Dinner     2
11         35.26  5.00  Female     No   Sun  Dinner     4
12         15.42  1.57    Male     No   Sun  Dinner     2
13         18.43  3.00    Male     No   Sun  Dinner     4
14         14.83  3.02  Female     No   Sun  Dinner     2
15         21.58  3.92    Male     No   Sun  Dinner     2
16         10.33  1.67  Female     No   Sun  Dinner     3
17         16.29  3.71    Male     No   Sun  Dinner     3
18         16.97  3.50  Female     No   Sun  Dinner     3
19         20.65  3.35    Male     No   Sat  Dinner     3
20         17.92  4.08    Male     No   Sat  Dinner     2
21         20.29  2.75  Female     No   Sat  Dinner     2
22         15.77  2.23  Female     No   Sat  Dinner     2
23         39.42  7.58    Male     No   Sat  Dinner     4
24         19.82  3.18    Male     No   Sat  Dinner     2
25         17.81  2.34    Male     No   Sat  Dinner     4
26         13.37  2.00    Male     No   Sat  Dinner     2
27         12.69  2.00    Male     No   Sat  Dinner     2
28         21.70  4.30    Male     No   Sat  Dinner     2
29         19.65  3.00  Female     No   Sat  Dinner     2
..           ...   ...     ...    ...   ...     ...   ...
214        28.17  6.50  Female    Yes   Sat  Dinner     3
215        12.90  1.10  Female    Yes   Sat  Dinner     2
216        28.15  3.00    Male    Yes   Sat  Dinner     5
217        11.59  1.50    Male    Yes   Sat  Dinner     2
218         7.74  1.44    Male    Yes   Sat  Dinner     2
219        30.14  3.09  Female    Yes   Sat  Dinner     4
220        12.16  2.20    Male    Yes   Fri   Lunch     2
```

In [18]:
```
# Not many 1- and 6-person parties
party_counts = party_counts.loc[:, 2:5]
party_counts
```

Out[18]:

size	2	3	4	5
day				
Fri	16	1	1	0
Sat	53	18	13	1
Sun	39	15	18	3
Thur	48	4	5	1

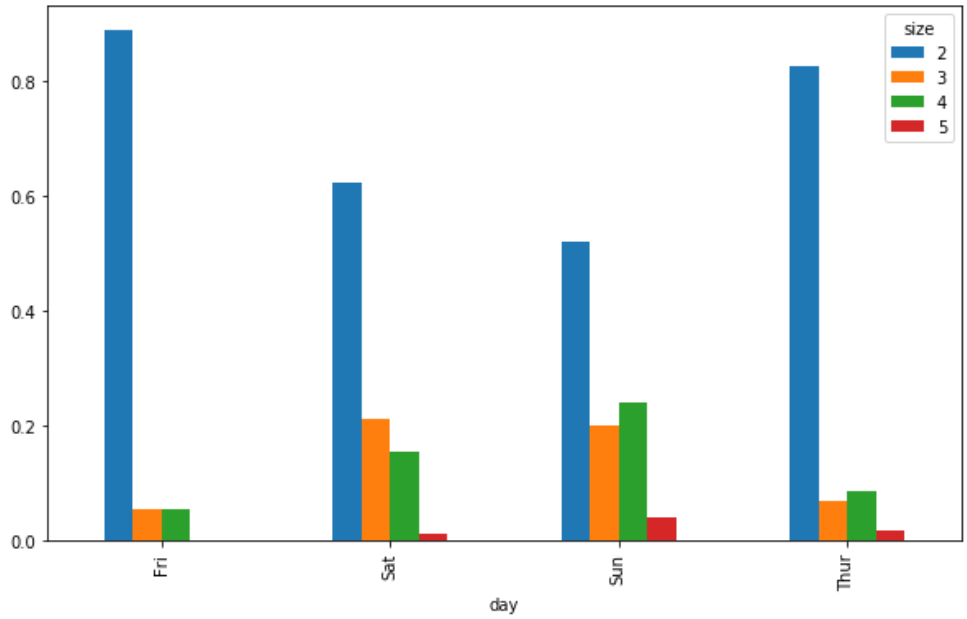
In [19]:
```
# Normalize to sum to 1
party_pcts = party_counts.div(party_counts.sum(1).astype(float), axi
s=0)
party_pcts
```

Out[19]:

size	2	3	4	5
day				
Fri	0.888889	0.055556	0.055556	0.000000
Sat	0.623529	0.211765	0.152941	0.011765
Sun	0.520000	0.200000	0.240000	0.040000
Thur	0.827586	0.068966	0.086207	0.017241

In [20]:
```
party_pcts.plot.bar()
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x117a4d748>
```



Histograms and Density Plots

The obj.hist() function for pandas objects is used to plot histograms.

- Notice, for categorical data, a value_counts will be needed to obtain the counts for hist()

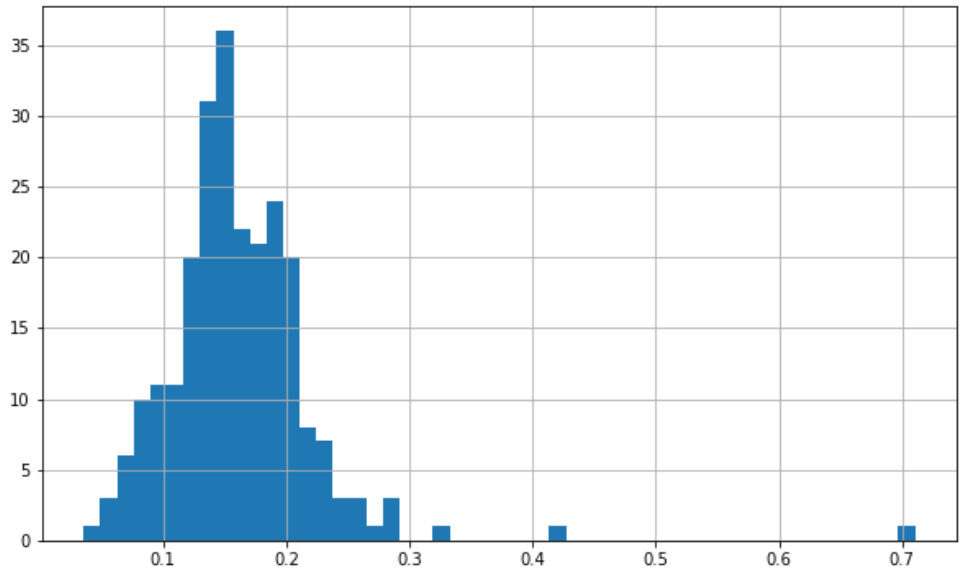
In [21]:
```
plt.figure()
```

Out[21]:
```
<Figure size 720x432 with 0 Axes>
```

```
<Figure size 720x432 with 0 Axes>
```

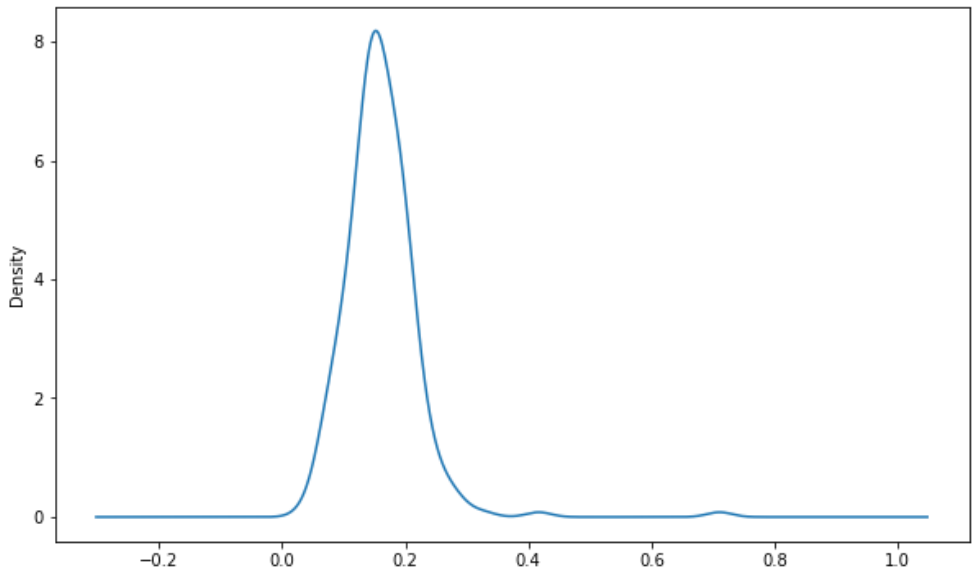
In [22]:
```
tips['tip_pct'] = tips['tip'] / tips['total_bill']
tips['tip_pct'].hist(bins=50)
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x117bd4160>



In [23]:
```
tips['tip_pct'].plot(kind='kde')
```

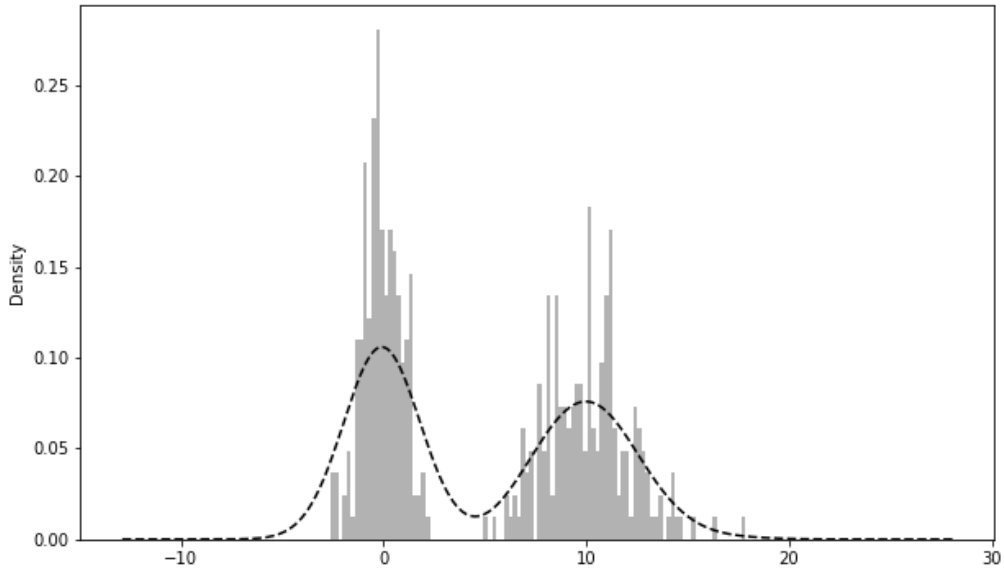
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x117c50e80>



In [24]:
```
comp1 = np.random.normal(0, 1, size=200)  # N(0, 1)
comp2 = np.random.normal(10, 2, size=200)  # N(10, 4)
values = Series(np.concatenate([comp1, comp2]))
values.hist(bins=100, alpha=0.3, color='k', normed=True)
values.plot(kind='kde', style='k--')
```

```
/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:
6462: UserWarning: The 'normed' kwarg is deprecated, and has bee
n replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been
"
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x1a199e8630>



Scatter Plots

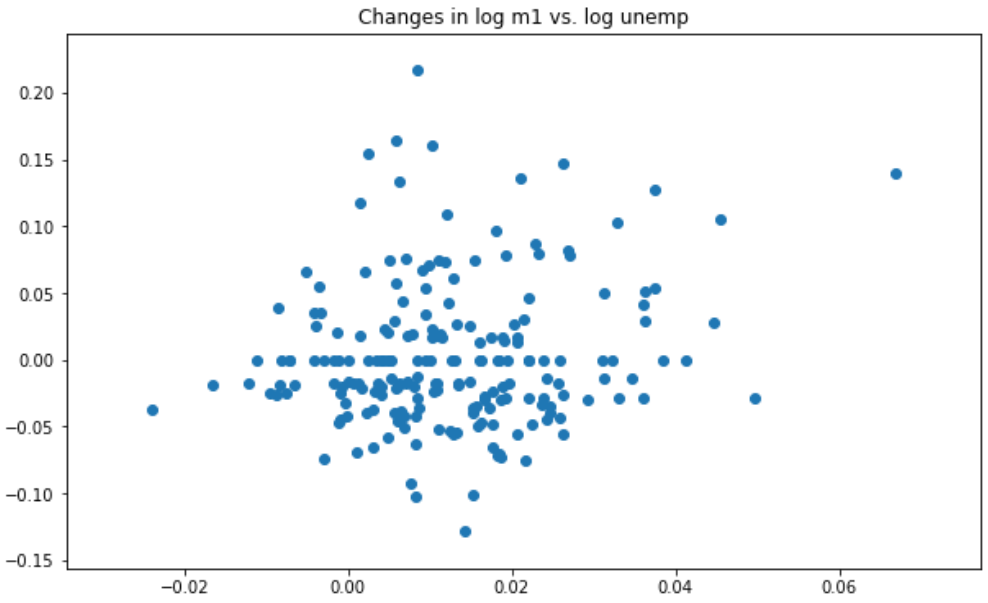
In [25]:
```
macro = pd.read_csv('ch08/macrodata.csv')
data = macro[['cpi', 'm1', 'tbilrate', 'unemp']]
trans_data = np.log(data).diff().dropna()
trans_data[-5:]
```

Out[25]:

	cpi	m1	tbilrate	unemp
198	-0.007904	0.045361	-0.396881	0.105361
199	-0.021979	0.066753	-2.277267	0.139762
200	0.002340	0.010286	0.606136	0.160343
201	0.008419	0.037461	-0.200671	0.127339
202	0.008894	0.012202	-0.405465	0.042560

In [26]:
```
plt.scatter(trans_data['m1'], trans_data['unemp'])
plt.title('Changes in log %s vs. log %s' % ('m1', 'unemp'))
```

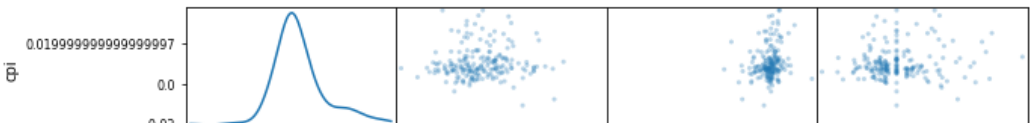
Out[26]: Text(0.5,1,'Changes in log m1 vs. log unemp')



In [27]:
```
#pd.scatter_matrix(trans_data, diagonal='kde', color='k', alpha=0.3)
pd.scatter_matrix(trans_data, diagonal='kde', alpha=0.3)
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2:
FutureWarning: pandas.scatter_matrix is deprecated, use pandas.p
lotting.scatter_matrix instead
```

Out[27]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a19d
cafd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a19e
06a90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a19f
19160>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a19f
407f0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a19f
69e80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a19f
69eb8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a19f
c2be0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a19f
f32b0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a1a0
1a940>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1a0
44fd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1a0
766a0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1a0
9dd30>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a1a0
d0400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1a0
f6a90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1a1
2a160>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1a1
507f0>]],
      dtype=object)
```



Texas Campaingn Finance 2018

Data from data.texas.gov

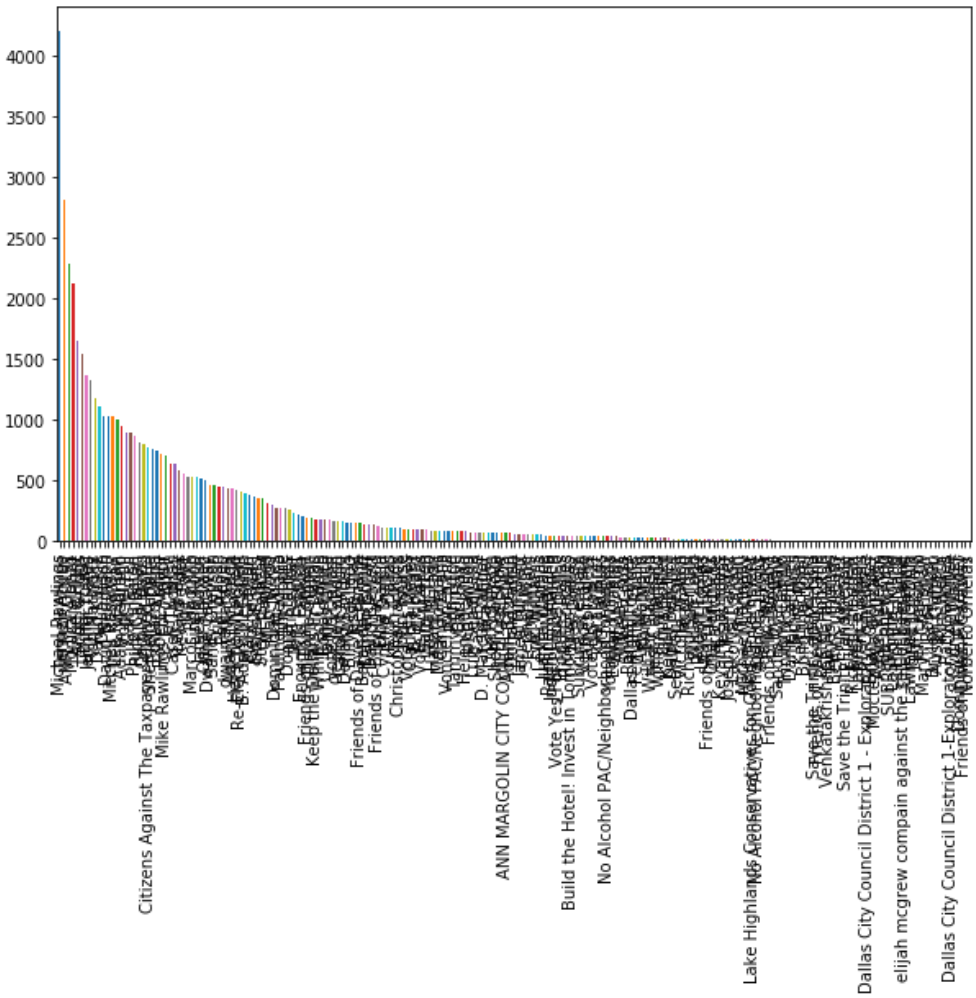
In [28]:
```
df = pd.read_csv('data/data_Texas_Campaign_Finance_2018.csv')
df[:5]
```

Out[28]:

	ID	Record ID	Report ID	File Link	Firs Nam
0	FR11598	1598	1598	http://campfin.dallascityhall.com /FinalReports...	Lester
1	221430	21430	1285	http://campfin.dallascityhall.com /FinalReports...	Kaye
2	221431	21431	1285	http://campfin.dallascityhall.com /FinalReports...	Mitche
3	221432	21432	1285	http://campfin.dallascityhall.com /FinalReports...	Robert
4	FR11601	1601	1601	http://campfin.dallascityhall.com /FinalReports...	Sandy

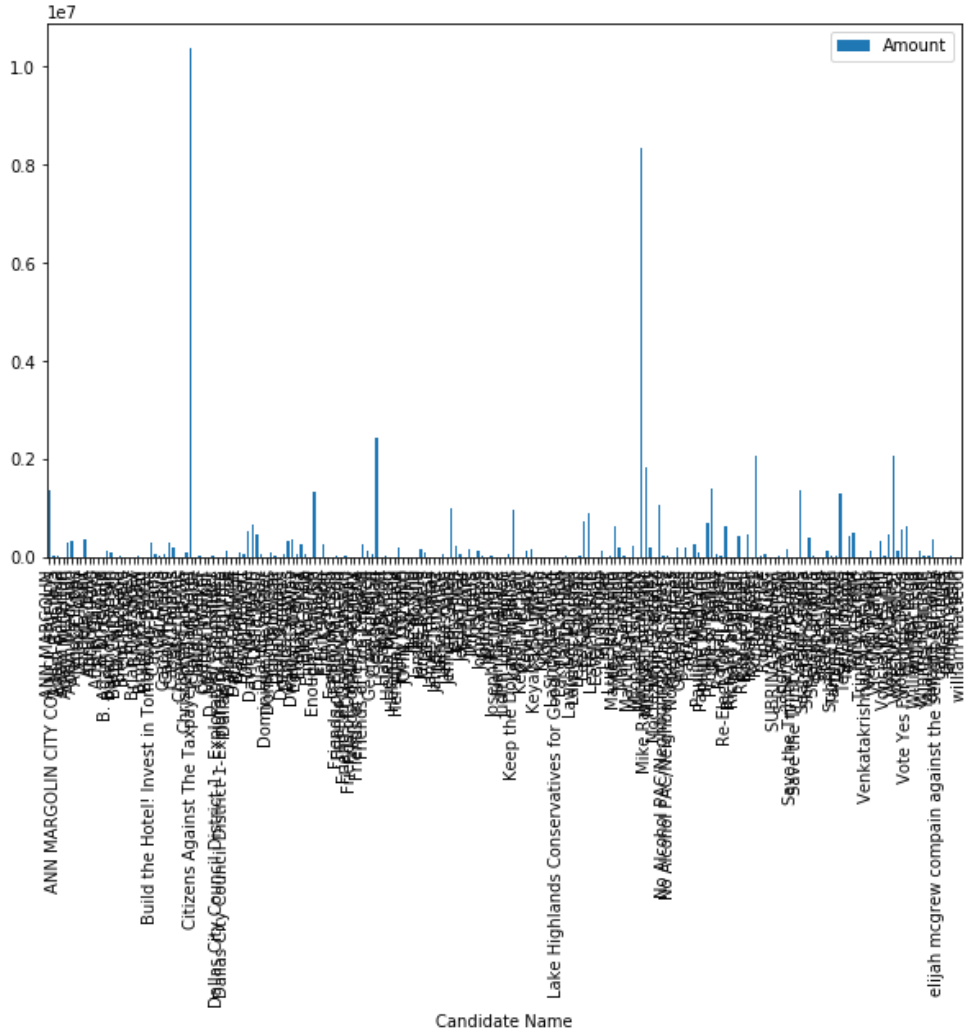
In [29]:
```
numContrib = df['Candidate Name'].value_counts()
numContrib.plot(kind='bar')
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1b518e80>



In [30]:
```
contrib = df[['Amount', 'Candidate Name']].groupby(
                'Candidate Name').aggregate('sum')
contrib.plot(kind='bar')
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x114ffba90>



Visualize Geographic Data on Maps

- Matplotlib uses Basemap object to visualize geographic data. The Basemap class is within mpl_toolkits package
- If Basemap is not found, run the following command in a terminal window:
  - conda install -c conda-forge basemap basemap-data-hires
- A Basemap object maintains information about a map, including types of earth project, direction, area, distance, shape, latitute, longitive, etc.
- The Basemap package contains a range of useful functions for drawing borders of physical features like continents, oceans, lakes, and rivers, as well as political boundaries such as countries and US states and counties.

Specific Methods to Place Data on Map

- contour()/contourf() : Draw contour lines or filled contours
- imshow(): Draw an image
- pcolor()/pcolormesh() : Draw a pseudocolor plot for irregular/regular meshes
- plot(): Draw lines and/or markers.
- scatter(): Draw points with markers.
- quiver(): Draw vectors.
- barbs(): Draw wind barbs.
- drawgreatcircle(): Draw a great circle.

EX: Visualizing Haiti Earthquake Crisis data

- The data is in a csv file and the map is in data files from basemap package

In [31]:
```
data = pd.read_csv('ch08/Haiti.csv')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3593 entries, 0 to 3592
Data columns (total 10 columns):
Serial             3593 non-null int64
INCIDENT TITLE     3593 non-null object
INCIDENT DATE      3593 non-null object
LOCATION           3592 non-null object
DESCRIPTION        3593 non-null object
CATEGORY           3587 non-null object
LATITUDE           3593 non-null float64
LONGITUDE          3593 non-null float64
APPROVED           3593 non-null object
VERIFIED           3593 non-null object
dtypes: float64(2), int64(1), object(7)
memory usage: 280.8+ KB
```

In [32]:
```
data[['INCIDENT DATE', 'LATITUDE', 'LONGITUDE']][:10]
```

Out[32]:

	INCIDENT DATE	LATITUDE	LONGITUDE
0	05/07/2010 17:26	18.233333	-72.533333
1	28/06/2010 23:06	50.226029	5.729886
2	24/06/2010 16:21	22.278381	114.174287
3	20/06/2010 21:59	44.407062	8.933989
4	18/05/2010 16:26	18.571084	-72.334671
5	26/04/2010 13:14	18.593707	-72.310079
6	26/04/2010 14:19	18.482800	-73.638800
7	26/04/2010 14:27	18.415000	-73.195000
8	15/03/2010 10:58	18.517443	-72.236841
9	15/03/2010 11:00	18.547790	-72.410010

In [33]:
```
data['CATEGORY'][:6]
```

Out[33]:
```
0          1. Urgences | Emergency, 3. Public Health,
1    1. Urgences | Emergency, 2. Urgences logistiqu...
2    2. Urgences logistiques | Vital Lines, 8. Autr...
3                             1. Urgences | Emergency,
4                             1. Urgences | Emergency,
5                        5e. Communication lines down,
Name: CATEGORY, dtype: object
```

In [34]:
```
data.describe()
```

Out[34]:

	Serial	LATITUDE	LONGITUDE
count	3593.000000	3593.000000	3593.000000
mean	2080.277484	18.611495	-72.322680
std	1171.100360	0.738572	3.650776
min	4.000000	18.041313	-74.452757
25%	1074.000000	18.524070	-72.417500
50%	2163.000000	18.539269	-72.335000
75%	3088.000000	18.561820	-72.293570
max	4052.000000	50.226029	114.174287

In [35]:
```
data = data[(data.LATITUDE > 18) & (data.LATITUDE < 20) &
            (data.LONGITUDE > -75) & (data.LONGITUDE < -70)
            & data.CATEGORY.notnull()]
data[:][:10]
```

Out[35]:

	Serial	INCIDENT TITLE	INCIDENT DATE	LOCATION	DESCRIPTIOI
0	4052	* URGENT * Type O blood donations needed in #J...	05/07/2010 17:26	Jacmel, Haiti	Birthing Clinic in Jacmel #Haiti urgently need...
4	4042	Citi Soleil school	18/05/2010 16:26	Citi Soleil, Haiti	We are workin with Haitian (NGO) -The Christi...
5	4041	Radio Commerce in Sarthe	26/04/2010 13:14	Radio Commerce Shelter, Sarthe	i'm Louinel from Sarthe. I'd to know what can ...
6	4040	Contaminated water in Baraderes.	26/04/2010 14:19	Marc near Baraderes	How do we treat water in areas without Pipe?\t...
7	4039	Violence at "arcahaie bas Saint-Ard"	26/04/2010 14:27	unable to find "arcahaie bas Saint-Ard&qu...	Goodnight at (arcahaie bas Saint-Ard) 2 young ...
8	4038	No electricity in pernier	15/03/2010 10:58	Pernier	why the peopl who lives in pernier doesn' fi...
9	4037	Shelter and food needed at Lamentin 54 and Rue	15/03/2010 11:00	Intersection of Lamentin 54 and Rue St Juste	GOOD EVENING ONG, I'M VERY HAPPY FOR THE AID

In [37]:
```
def to_cat_list(catstr):
    stripped = (x.strip() for x in catstr.split(','))
    return [x for x in stripped if x]

def get_all_categories(cat_series):
    cat_sets = (set(to_cat_list(x)) for x in cat_series)
    return sorted(set.union(*cat_sets))

def get_english(cat):
    code, names = cat.split('.')
    if '|' in names:
        names = names.split(' | ')[1]
    return code, names.strip()
```

In [38]:
```
get_english('2. Urgences logistiques | Vital Lines')
```

Out[38]:
```
('2', 'Vital Lines')
```

In [39]:
```
all_cats = get_all_categories(data.CATEGORY)
# Generator expression
english_mapping = dict(get_english(x) for x in all_cats)
english_mapping['2a']
english_mapping['6c']
```

Out[39]:
```
'Earthquake and aftershocks'
```

In [41]:
```
def get_code(seq):
    return [x.split('.')[0] for x in seq if x]

all_codes = get_code(all_cats)
code_index = pd.Index(np.unique(all_codes))
dummy_frame = DataFrame(np.zeros((len(data), len(code_index))),
                        index=data.index, columns=code_index)
```

In [42]:
```
dummy_frame.iloc[:, :6].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3569 entries, 0 to 3592
Data columns (total 6 columns):
1     3569 non-null float64
1a    3569 non-null float64
1b    3569 non-null float64
1c    3569 non-null float64
1d    3569 non-null float64
2     3569 non-null float64
dtypes: float64(6)
memory usage: 195.2 KB
```

In [43]:
```
for row, cat in zip(data.index, data.CATEGORY):
    codes = get_code(to_cat_list(cat))
    dummy_frame.ix[row, codes] = 1

data = data.join(dummy_frame.add_prefix('category_'))
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3:
DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-ind
exer-is-deprecated
  This is separate from the ipykernel package so we can avoid do
ing imports until
```

In [44]:
```
data.iloc[:, 10:15].info()
```

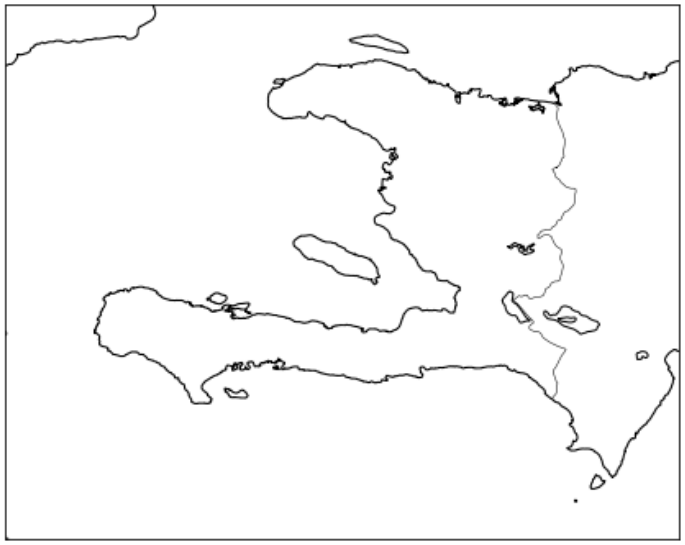
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3569 entries, 0 to 3592
Data columns (total 5 columns):
category_1     3569 non-null float64
category_1a    3569 non-null float64
category_1b    3569 non-null float64
category_1c    3569 non-null float64
category_1d    3569 non-null float64
dtypes: float64(5)
memory usage: 327.3 KB
```

Plot the Map

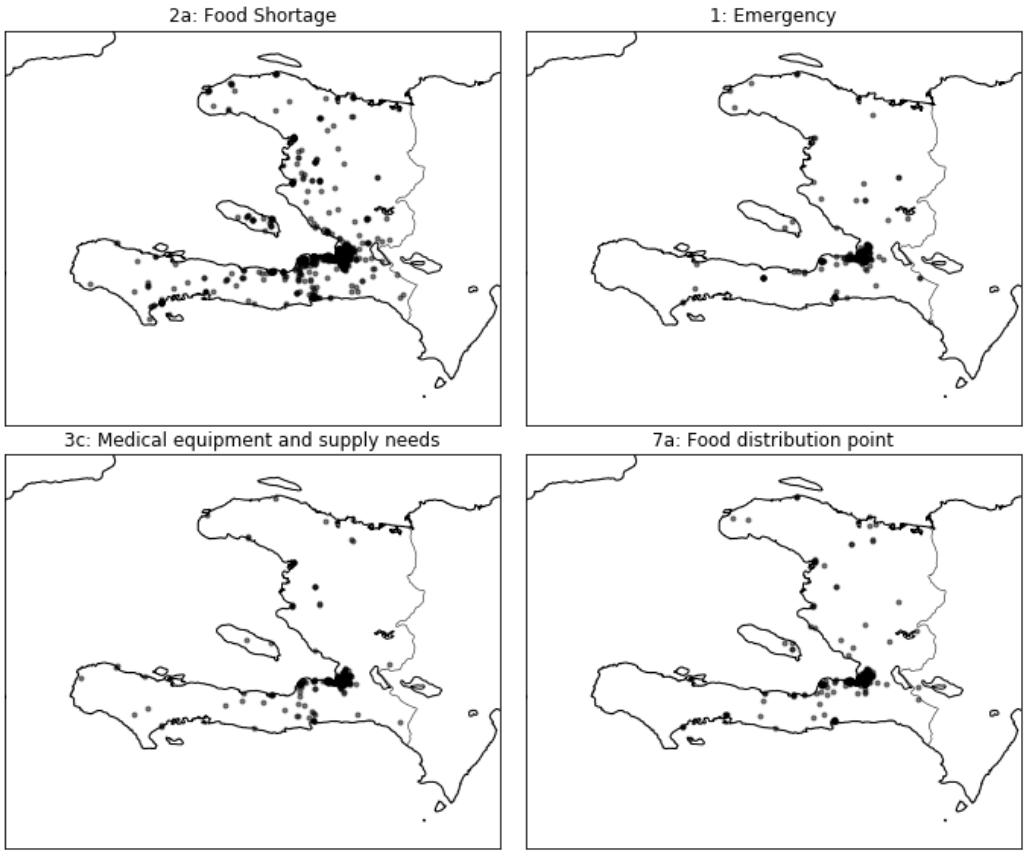
This will take some time to draw ...

In [45]:
```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

def basic_haiti_map(ax=None, lllat=17.25, urlat=20.25,
                    lllon=-75, urlon=-71):
    # create polar stereographic Basemap instance.
    m = Basemap(ax=ax, projection='stere',
                lon_0=(urlon + lllon) / 2,
                lat_0=(urlat + lllat) / 2,
                llcrnrlat=lllat, urcrnrlat=urlat,
                llcrnrlon=lllon, urcrnrlon=urlon,
                resolution='f')
    # draw coastlines, state and country boundaries, edge of map.
    m.drawcoastlines()
    m.drawstates()
    m.drawcountries()

    return m

p = basic_haiti_map()
p.plot(100, 100, 'k.', alpha=0.5)
```

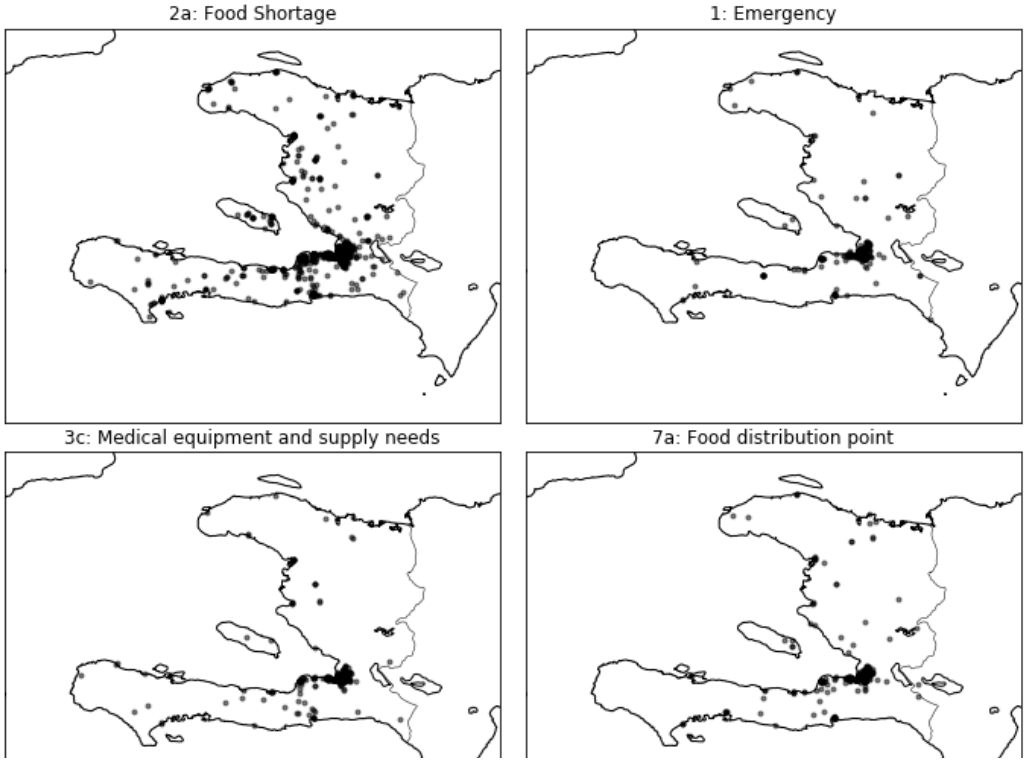
Out[45]: [<matplotlib.lines.Line2D at 0x1a1c0db588>]



In [46]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))
fig.subplots_adjust(hspace=0.05, wspace=0.05)

to_plot = ['2a', '1', '3c', '7a']

lllat=17.25; urlat=20.25; lllon=-75; urlon=-71

for code, ax in zip(to_plot, axes.flat):
    m = basic_haiti_map(ax, lllat=lllat, urlat=urlat,
                        lllon=lllon, urlon=urlon)

    cat_data = data[data['category_%s' % code] == 1]

    # compute map proj coordinates.
    x, y = m(cat_data.LONGITUDE.values, cat_data.LATITUDE.values)

    m.plot(x, y, 'k.', alpha=0.5)
    ax.set_title('%s: %s' % (code, english_mapping[code]))
```



In [47]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))
fig.subplots_adjust(hspace=0.05, wspace=0.05)

to_plot = ['2a', '1', '3c', '7a']

lllat=17.25; urlat=20.25; lllon=-75; urlon=-71

def make_plot():

    for i, code in enumerate(to_plot):
        cat_data = data[data['category_%s' % code] == 1]
        lons, lats = cat_data.LONGITUDE, cat_data.LATITUDE

        ax = axes.flat[i]
        m = basic_haiti_map(ax, lllat=lllat, urlat=urlat,
                            lllon=lllon, urlon=urlon)

        # compute map proj coordinates.
        x, y = m(lons.values, lats.values)

        m.plot(x, y, 'k.', alpha=0.5)
        ax.set_title('%s: %s' % (code, english_mapping[code]))


make_plot()
```



In [48]:
```
shapefile_path = 'ch08/PortAuPrince_Roads/PortAuPrince_Roads'
m.readshapefile(shapefile_path, 'roads')
```

Out[48]:
```
(1583,
 3,
 [-72.749246, 18.409952, 0.0, 0.0],
 [-71.973789, 18.7147105, 0.0, 0.0],
 <matplotlib.collections.LineCollection at 0x1a1d8c3860>)
```